

FIG. 1

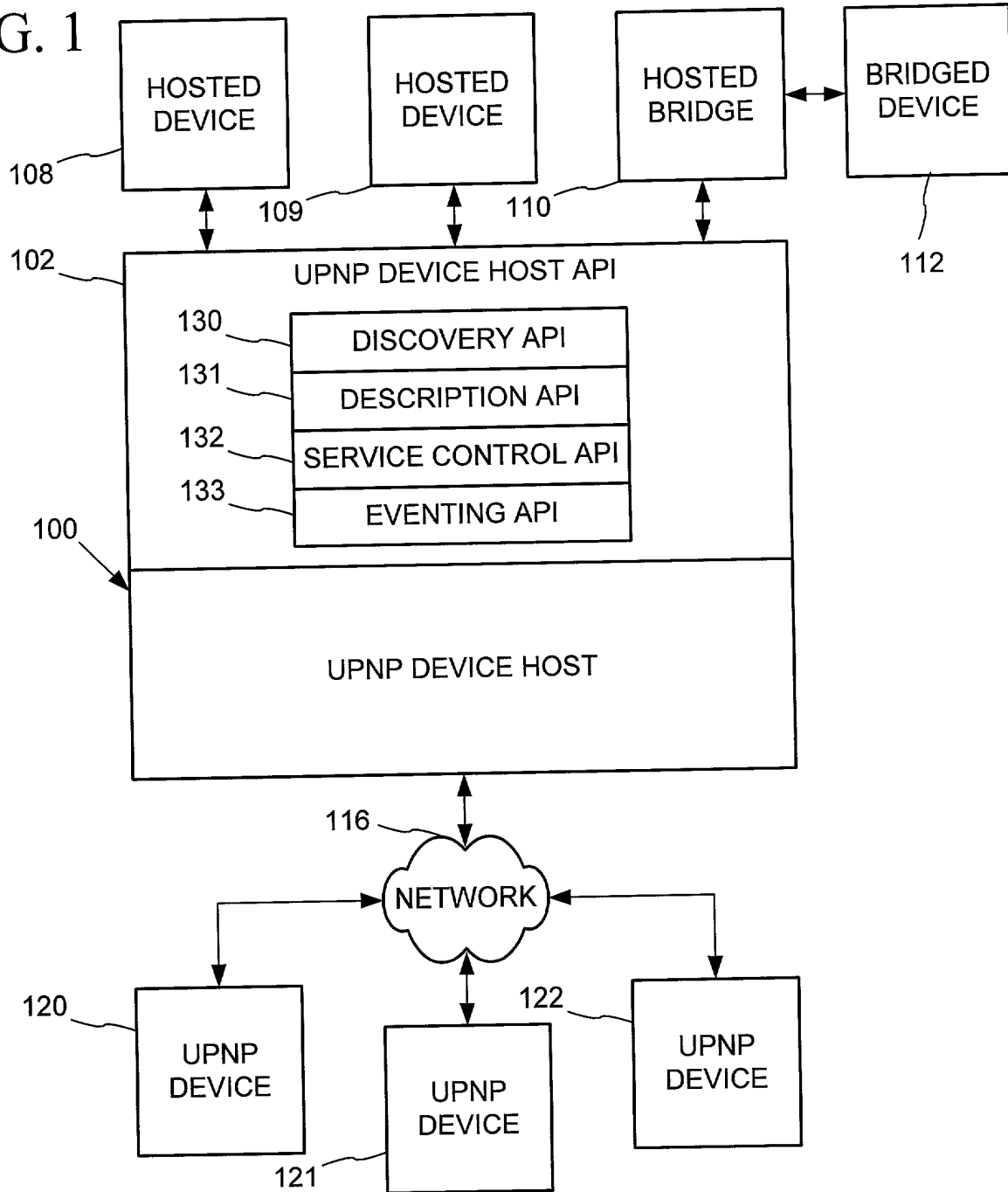


FIG. 1

FIG. 2

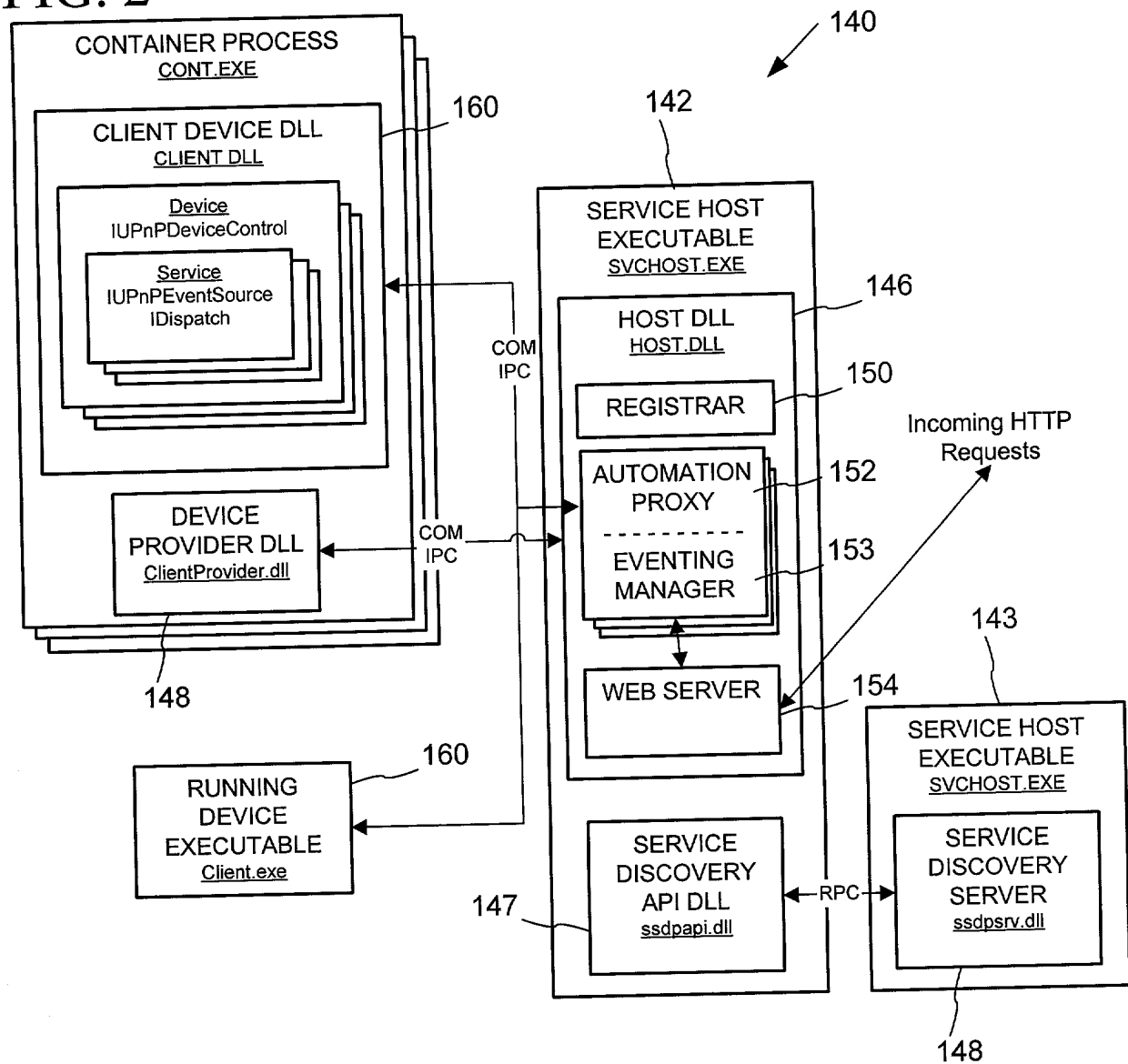


FIG. 3

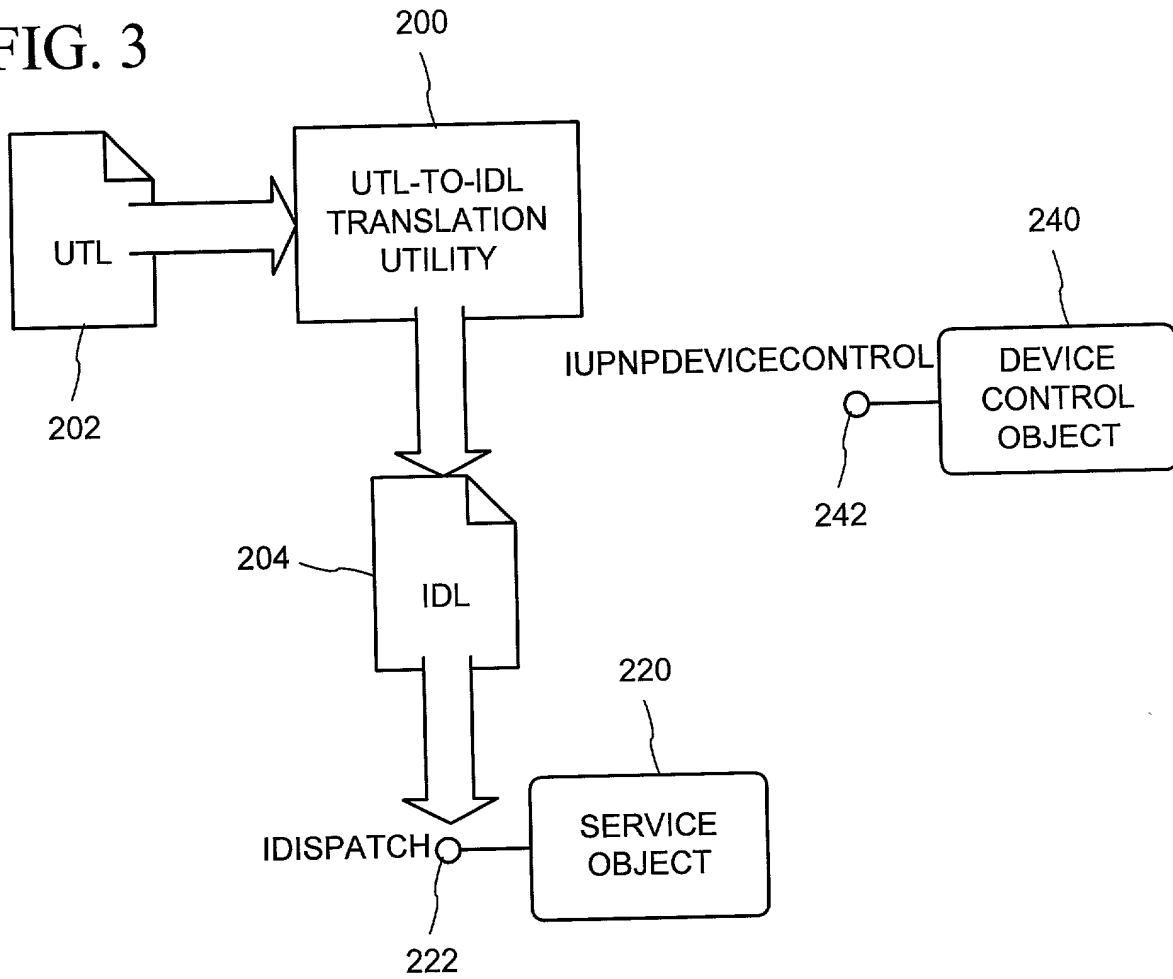


FIG. 4

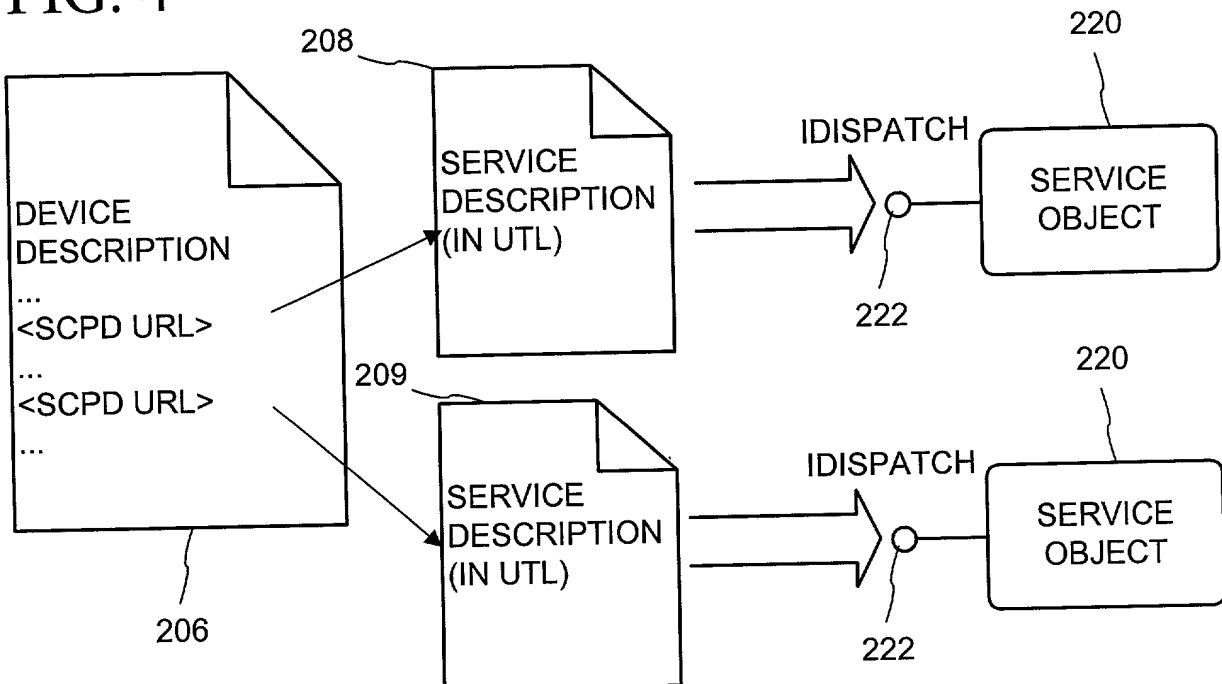


FIG. 5

```
<?xml version="1.0"?>
<scpd xmlns="x-schema:UTL-schema.xml">
  <serviceStateTable>
    <stateVariable>
      <name>Chapter</name>
      <dataType>i4</dataType>
      <allowedValueRange>
        <minimum>1</minimum>
        <maximum>50</maximum>
        <step>1</step>
      </allowedValueRange>
      <defaultValue>0</defaultValue>
    </stateVariable>
  </serviceStateTable>

  <actionList>
    <action>
      <name>GotoChapter</name>
      <argumentList>
        <argument>
          <name>NewChapter</name>
          <direction>in</direction>
          <relatedStateVariable>Chapter</relatedStateVariable>
        </argument>
        <argument>
          <name>OldChapter</name>
          <direction>out</direction>
          <relatedStateVariable>Chapter</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
  </actionList>
</scpd>
```

FIG. 6

```
[
  object
  uuid(ad8b5af4-93fe-4f21-8ff9-a569dc8fcf26),
  helpstring("avtransport-0-1 interface"),
  oleautomation
]
interface IUPnPService_avtransport_0_1
{
  // State Variables (properties)
  [propget] HRESULT Chapter([out, retval] INT * pval);

  // Actions (methods)
  HRESULT GotoChapter ([in, out] INT * pChapter);
}

[
  uuid(8944228e-746b-439b-83ba-089d141eaf6f)
]
library UPNPPrivLib
{
  [
    uuid(7e46fad0-e36b-4be9-b7f2-55c8c075ee4a),
    helpstring("avtransport-0-1 dispinterface"),
    nonextensible
  ]
  dispinterface DIUPnPService_avtransport_0_1
  {
    interface IUPnPService_avtransport_0_1;
  };
};
```

FIG. 7

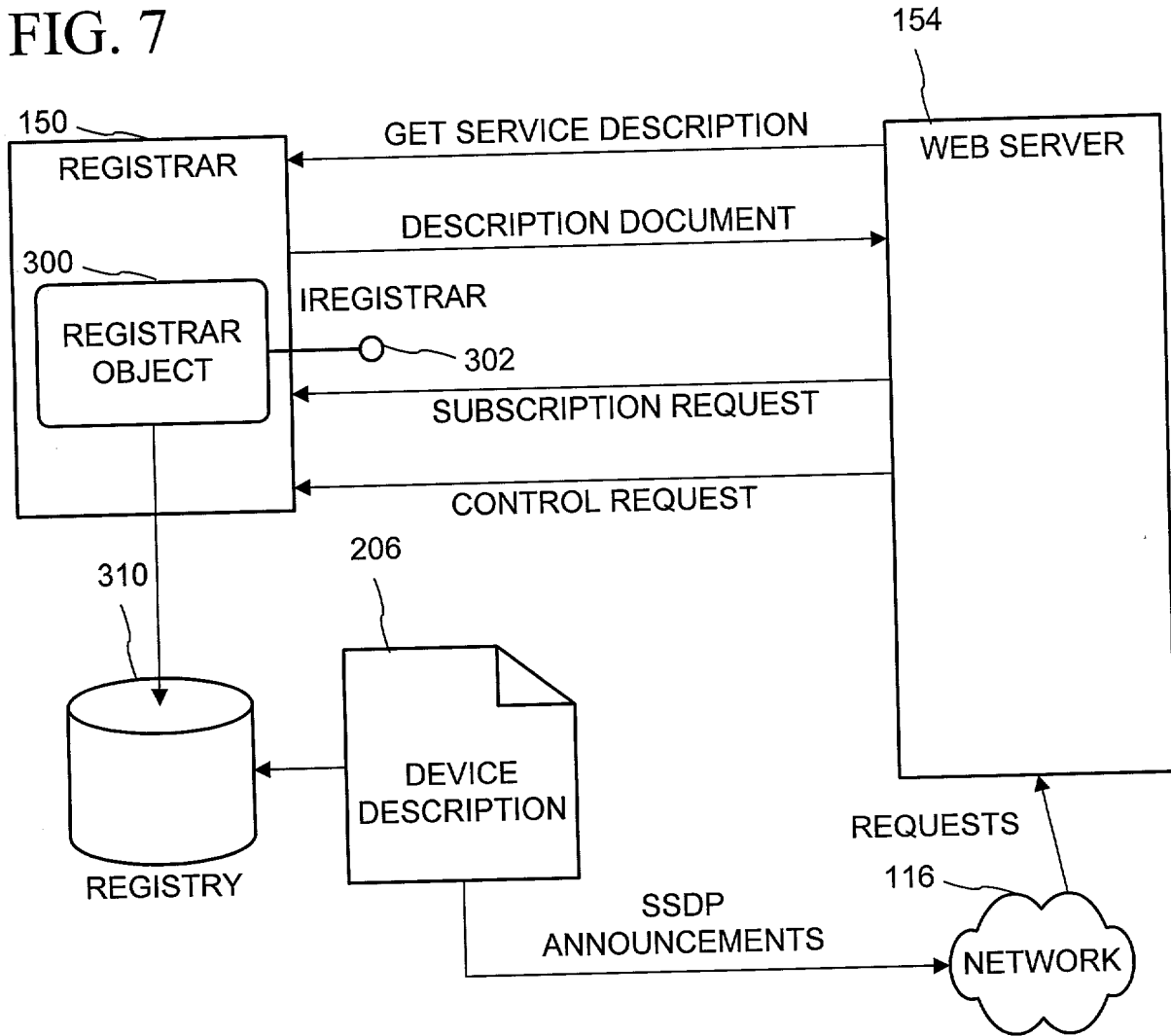


FIG. 8

```
[
    uuid(bebc3352-9eb8-4736-b1eb-093daa00d113),
    helpstring("UPnPRegistrar Class")
]
coclass UPnPRegistrar
{
    [default] interface IUPnPRegistrar;
    interface IUPnPReregistrar;
};
```

FIG. 9

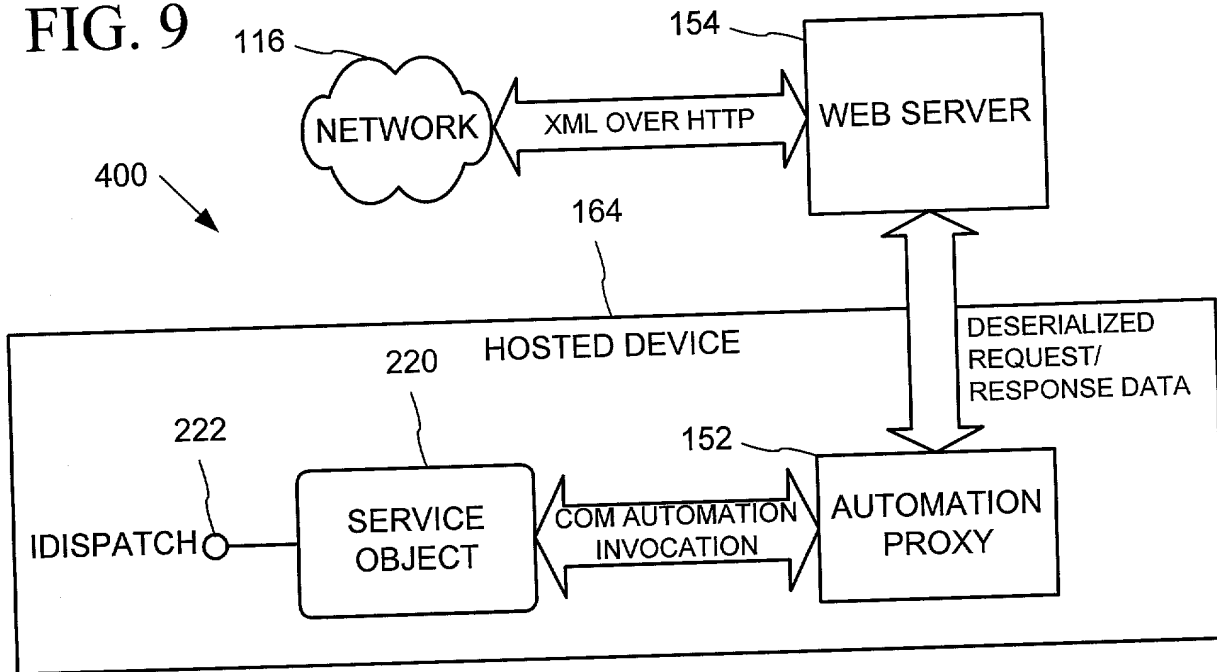


FIG. 9

FIG. 10

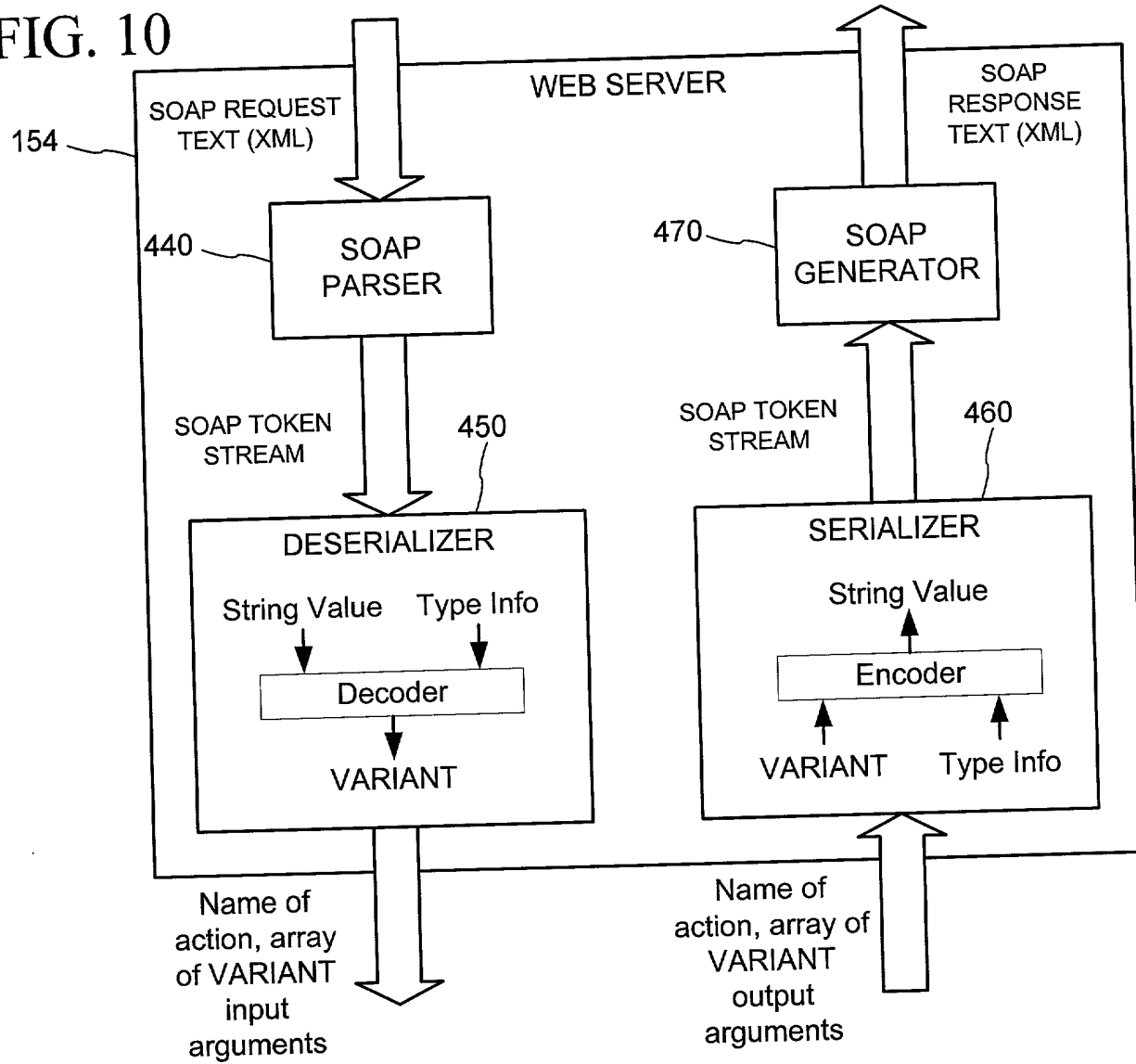


FIG. 11

```
struct tagUPNP_SOAP_REQUEST
{
    BSTR          bstrActionName
    IXMLDOMNodeList * pxdnlArgs
} UPNP_SOAP_REQUEST;
```

FIG. 12

```
struct tagUPNP_CONTROL_REQUEST
{
    BSTR          bstrActionName;
    DWORD         cInputArgs;
    VARIANT       rgvarInputArgs[];
} UPNP_CONTROL_REQUEST;
```

FIG. 13

```
struct tagUPNP_CONTROL_RESPONSE
{
    BSTR bstrActionName;
    union
    {
        struct
        {
            DWORD          cOutputArgs;
            VARIANT         rgvarOutputArgs[];
        };
        struct
        {
            BSTR           bstrFaultCode;
            BSTR           bstrFaultString;
            DWORD          dwUPnPErrorCode;
            BSTR           bstrUPnPErrorString;
        };
    };
} UPNP_CONTROL_RESPONSE;
```

FIG. 14

```
Deserialize(
    UPNP_SOAP_REQUEST      * pusr,
    IUPnPServiceDescriptionInfo * pusdi,
    UPNP_CONTROL_REQUEST    * pucr)
{
    IXMLDOMNode * pxdnArgument = NULL;

    // Set up UPNP_CONTROL_REQUEST structure.

    pucr->bstrActionName = SysAllocString(pusr->bstrActionName);
    pucr->clnputArgs = pusr->pxdnArgs->length;
    pucr->rgvarInputArgs = AllocateMemory(pucr->clnputArgs *
                                           sizeof(VARIANT));

    // Deserialize each argument

    DWORD i = 0;

    for each pxdnArgument in pusr->pxdnArgs
    {
        VARIANT varDataType;

        VariantInit(&varDataType);
        varDataType.vt = VT_BSTR;
        V_BSTR(&varDataType) =
            pusdi->GetArgumentType(pucr->bstrActionName,
                                   pxdnArgument->name);

        pxdnArgument->put_dataType(varDataType);
        pxdnArgument->get_nodeTypedValue(&pucr->rgvarInputArgs[i]);
        i++;
    }
}
```

FIG. 15

```

struct tagUPNP_SOAP_RESPONSE
{
    union
    {
        struct
        {
            BSTR      bstrActionName;
            IXMLDOMNodeList * pxdnlOutArgs;
        };
        struct
        {
            BSTR      bstrFaultCode;
            BSTR      bstrFaultString;
            BSTR      bstrUPnPErrorCode;
            BSTR      bstrUPnPErrorMessage;
        };
    }
} UPNP_SOAP_RESPONSE;

```

FIG. 16

```

[
    uuid(0FB40F0D-1021-4022-8DA0-AAB0588DFC8B),
    helpstring("AutomationProxy Class")
]
coclass UPnPAutomationProxy
{
    [default] interface IUPnPAutomationProxy;
    interface IUPnPServiceDescriptionInfo;
};

```

FIG. 17

```
[
    object,
    uuid(de2eca21-99d0-4d92-b2ae-cb994e85f31f),
    pointer_default(unique),
    version(1.0)
]

interface IUPnPEventManager : IUnknown
{
    [helpstring("method Initialize")]
    HRESULT Initialize(
        [in] LPCWSTR          szUdn,
        [in] LPCWSTR          szSid,
        [in] IUPnPAutomationProxy * puap,
        [in] IUnknown *      punkSvc);
    [helpstring("method AddSubscriber")]
    HRESULT AddSubscriber(
        [in] LPWSTR  szCallbackUrl,
        [in] DWORD * pcsecTimeout,
        [out] LPWSTR * pszSid);
    [helpstring("method RenewSubscriber")]
    HRESULT RenewSubscriber(
        [in] DWORD * pcsecTimeout,
        [in] LPWSTR  szSid);
    [helpstring("method RemoveSubscriber")]
    HRESULT RemoveSubscriber(
        [in] LPWSTR  szSid);
    [helpstring("method Shutdown")]
    HRESULT Shutdown();
}

interface IUPnPEventSink: IUnknown
{
    [helpstring("method OnStateChanged"), hidden]
    HRESULT OnStateChanged(
        [in] DWORD cChanges,
        [in, size_is(cChanges)] DISPID rgdispidChanges[]);
    [helpstring("method OnStateChangedSafe")]
    HRESULT OnStateChangedSafe(
        [in] SAFEARRAY(DISPID) rgdispidChanges);
}
```

FIG. 18

```
[
    object,
    uuid(b05e0973-c342-4650-9d04-9224c6e6ded9),
    pointer_default(unique),
    version(1.0)
]

interface IUPnPEventSource: IUnknown
{
    [helpstring("method Advise")]
    HRESULT Advise(
        [in] IUnknown * punkSubscriber);

    [helpstring("method Unadvise")]
    HRESULT Unadvise(
        [in] IUnknown * punkSubscriber);
}
```

FIG. 19

```

struct UPNP_SUBSCRIBER
{
    UPNP_EVENT_SOURCE *    pes;           // Pointer to event source
    LPWSTR                 szUrl;         // Callback URL
    FILETIME                ftTimeout;    // Timeout period
    DWORD                  csecTimeout;   // Timeout period
    DWORD                  iSeq;          // Event sequence number
    LPTSTR                  szSid;        // Subscription Identifier
    DWORD                  cRenewals;     // # of renewals received
    HANDLE                  hEventQ;      // Event signaled when Q full
    UPNP_EVENT *           pueQueue;     // Event queue
    HANDLE                  hWait;        // Handle of registered wait
    UPNP_SUBSCRIBER *      psubNext;     // Next subscriber in list
};

struct UPNP_EVENT_SOURCE
{
    LPWSTR                 szEsid;        // Event source identifier
    UPNP_SUBSCRIBER *      pusList;       // List of subscribers
    HANDLE                  hTimerQ;      // Handle to timer queue
    UPNP_EVENT_SOURCE *    pesNext;       // Next in list
    DWORD                  csecTimeout;   // Preferred timeout
};

struct UPNP_RENEWAL
{
    LPWSTR                 szEsid;        // Event source identifier
    LPTSTR                  szSid;        // Subscription Identifier
    DWORD                  iRenewal;     // renewal index
};

struct UPNP_WAIT_PARAMS
{
    LPWSTR                 szEsid;        // Event source identifier
    LPTSTR                  szSid;        // Subscription Identifier
};

struct UPNP_EVENT
{
    LPWSTR                 szBody;        // body of event message
};

```


FIG. 20

```
HRESULT HrRegisterEventSource(LPCWSTR szEsid);

HRESULT HrDeregisterEventSource(LPCWSTR szEsid);

HRESULT HrAddSubscriber(
    LPCWSTR      szEsid,
    LPCWSTR      szCallbackUrl,
    LPCWSTR      szEventBody,
    DWORD *      pcsecTimeout,
    LPWSTR       * pszSid);

HRESULT HrRenewSubscriber(
    LPCWSTR      szEsid,
    DWORD *      pcsecTimeout,
    LPCWSTR      szSid);

HRESULT HrRemoveSubscriber(
    LPCWSTR      szEsid,
    LPCWSTR      szSid);

HRESULT HrSubmitEvent(LPCWSTR szEsid,
                      LPCWSTR szEventBody);

HRESULT HrSubmitEventZero(LPCWSTR szEsid, LPCWSTR szSid,
                          LPCWSTR szEventBody);
```

2023-11-06 10:10:11

FIG. 21

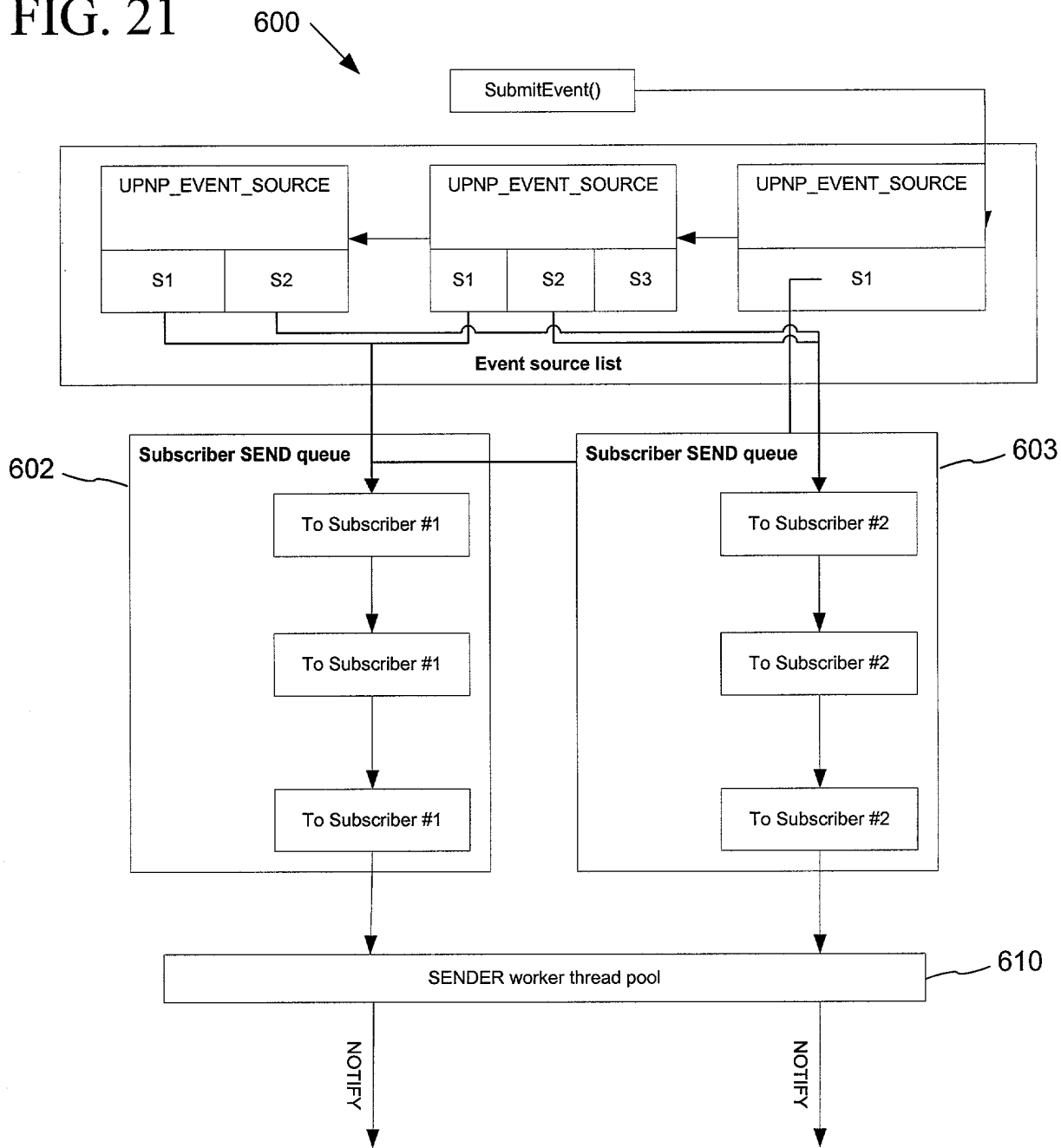


FIG. 22

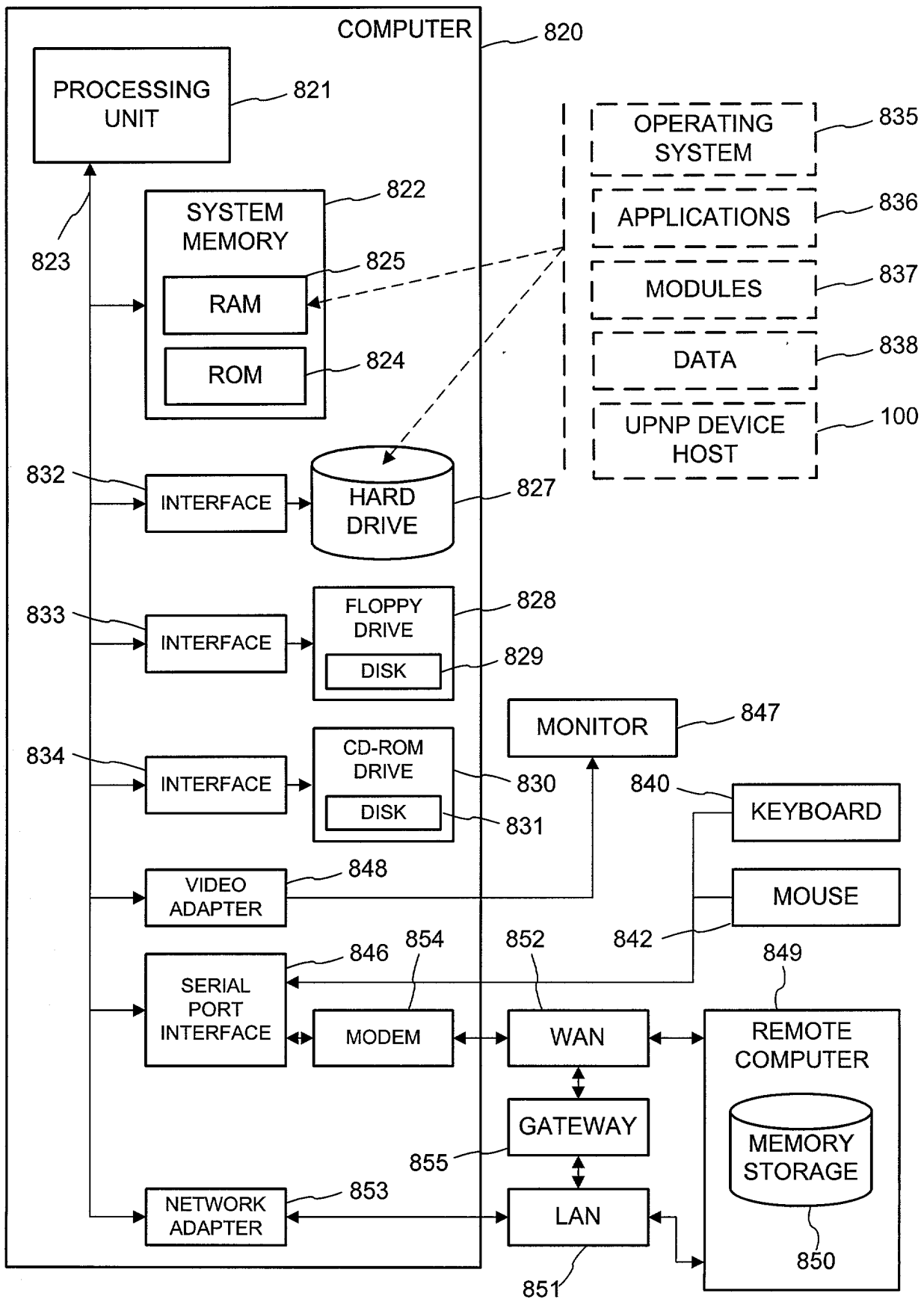


FIG. 23

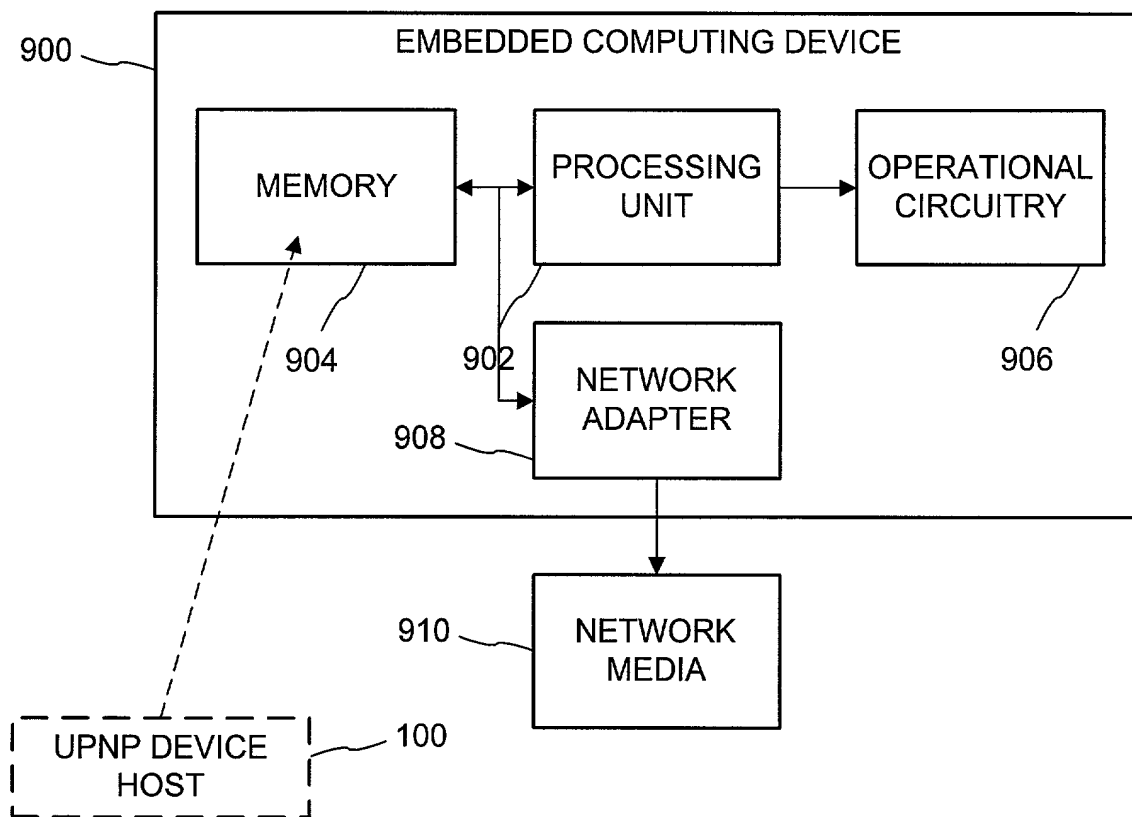


FIG. 24

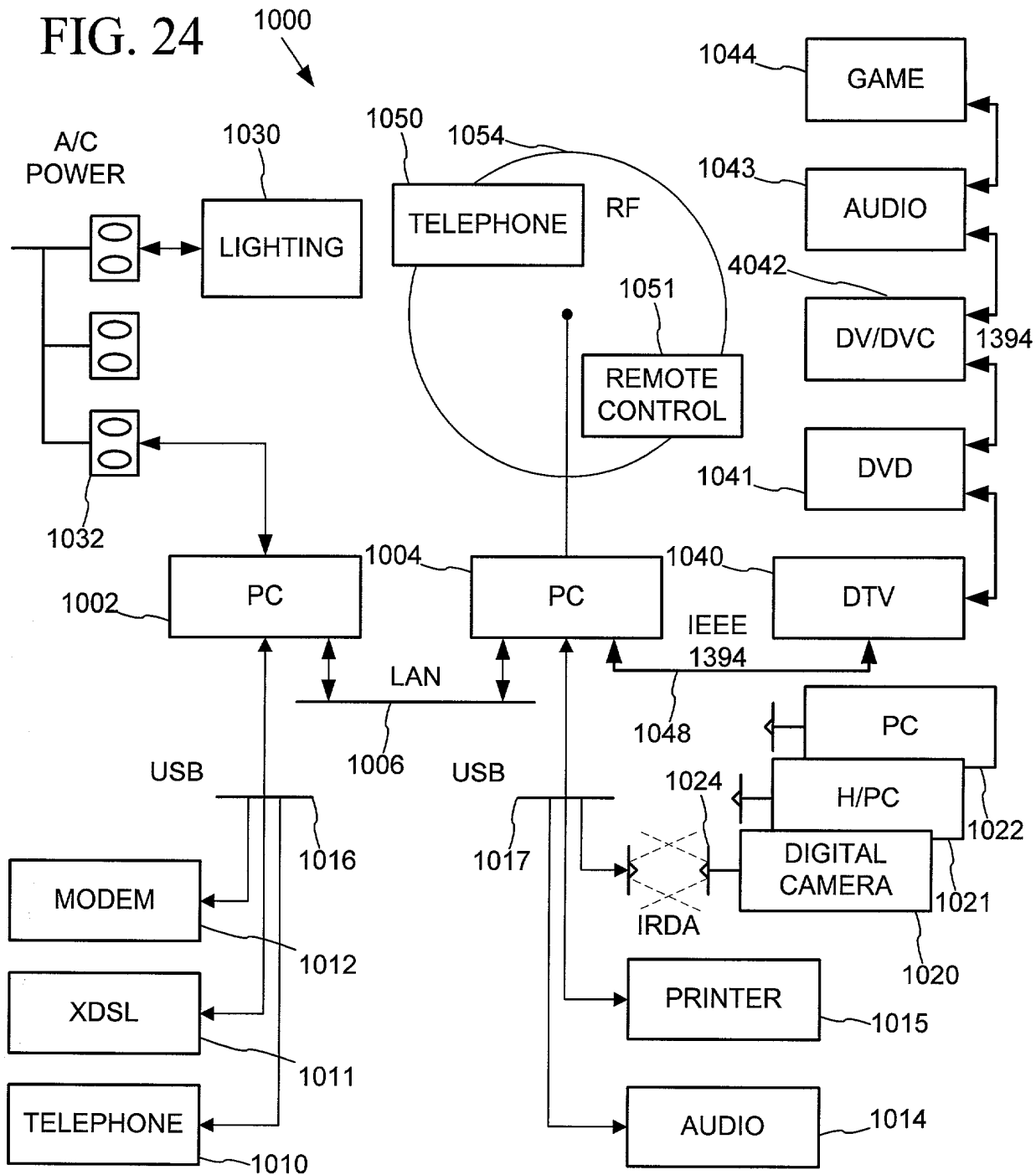


FIG. 24